Amendments to the Specification

Please replace the paragraph that begins on Page 16, line 18 and carries over to Page 17, line 10 with the following marked-up replacement paragraph:

-- To accomplish this, preferred embodiments define an instance identity as a single string. The root class in the managed object hierarchy, referred to herein as "Element", has an "InstanceID" property (or a property with some similar name) that is used as the key for all classes in the hierarchy. The Element hierarchy is single-rooted and uses single inheritance. (In a system that does not have a single root, an artificial single root can be created to support this approach.) Every instance of every class has a unique identity, which is stored [[at]] as the value of its InstanceID property. The InstanceID property's structure is defined such that applications that need to use a resource's identity, but do not need to know the structure of that identity, may treat it as an opaque value. However, the structure of the string is well-defined, so that clients, users, agents, and instrumentation that need access to the details of how a specific instance is identified may extract those details from the identity string. Subclassing is used to provide details of how the value of the InstanceID property is constructed for instances of that class. --

Please replace the paragraph on Page 25, lines 10 - 19 with the following marked-up replacement paragraph:

-- Turning now to a discussion of how instance identities are used to access data in repositories, it is known that long, variable-length keys are very inefficient, and that the most efficient keys are reasonable, short, fixed-length binary values. Therefore, to efficiently access data repositories that store managed resources, a deterministic technique may be used that

generates a binary/numeric key value from an ~~Instanced~~ InstanceID property's value. It is not necessary that this technique is reversible. That is, it is not necessary to be able to deduce the value of the managed object's ~~instanced~~ InstanceID property from its corresponding binary/numeric key value. Instead, given a particular binary key, the entire instance can be retrieved from the repository and the instance's identity can then be constructed from the naming rule for that instance's class and the instance's properties. --

Please replace the paragraph that begins on Page 29, line 18 and carries over to Page 30, line 16 with the following marked-up replacement paragraph:

-- The next naming rule illustrated in Fig. 2 is the naming rule 230 for instances of Application class 150. As shown in that rule 230, an instance of Application class 150 is to be identified using its ProcessID property, within the scoping context of the ComputerSystem class, to construct the identity of instances of this class. A particular computer system, in other words, may use the same name for its applications as are used by another computer system, but within that particular computer system, the application names are not reused. A sample instance of Application class 150 may have the property values illustrated at 500 in Fig. 5A, having "EMACS" as the value of its Name property and "16325" as the value of its ProcessID property. Since instances of Application class must be scoped by an instance of ComputerSystem class for uniqueness, and those instances must in turn be scoped by an instance of Organization class, and since Organization has a universally unique naming scheme, a root scope is not required in the identity of an Application instance but the domain name and host name of the scoping instances must be provided. For the sample instance 500 of Fig. 5A, the identity is therefore constructed as

shown at 550 in Fig. 5B. This identity 550 begins with the identity of the scoping context, which is a particular computer system ~~(within~~ (which, in turn, identifies its organization, as discussed above with reference to Fig. 4B), and is followed by the class name and a list of property name/value pairs that uniquely identify this instance of Application class. The class name of the present instance is "Application", and according to naming rule 230, this instance specifies the "ProcessID" property name and its value "16325", thereby providing the unique identity 550 corresponding to instance 500 of Fig. 5A. --

Please replace the paragraph on Page 32, lines 9 - 19 with the following marked-up replacement paragraph:

-- Control reaches Block 715 when all of the property name/value pairs have been added to the identity string. A check is then made to determine whether this instance's identity is defined within the scope of another object. (As described above with reference to Fig. 2, this scoping information is preferably specified with the naming rule for each class. Alternatively, "Element" class 110 of Fig. 1 may include a separate "ScopingContext" property with which each class may specify its scoping context for that class, if any.) If the test at Block 715 has a positive result, [[the]] then Block 720 prepends the identity of the scoping object to the instance's current identity (i.e., to the identity string which is being created). Control then returns to Block 715 to determine whether any further scoping context information is needed. (For example, as illustrated by identity 550 of Fig. 5B, a scoping context may be defined with reference to a class that has its own scoping context.) --